SERVER COMPUTER ISSUED CREDENTIAL AUTHENTICATION

FIELD OF THE INVENTION

[0001] The present invention generally relates to computer network security and more particularly to a system and method for providing a limited credential for authenticating access to network resources.

BACKGROUND OF THE INVENTION

[0002] In order to access network resources, a user enters identification and password information into a client computer that is transmitted to a server computer over a computer network for authentication. In turn, the server computer authenticates the client computer using the password and identification information, thereby allowing access to network resources. If the connection with the server computer is terminated, the user must re-enter the identification and password information into the client computer in order to re-authenticate and reconnect with the server computer. The user must re-enter the information because the password is not stored on the client computer for security reasons.

[0003] A "credential" can be issued to the client computer for facilitating the reconnect procedure. The credential is data that is used to prove the identity of the subject. In this instance, the credential is used by the client computer to authenticate the client computer to the server computer such that the user does not need to re-enter or store on the client computer the password information. There is no need for the client computer to transmit the identification and password information to the server

computer if the client computer has the credential because the credential ensures with a high degree of reliability that the client computer should have access.

[0004] The credential is generated by a trusted third party (TTP) such as the type used with the Kerberos system. The TTP issues the credential that allows the client computer to authenticate itself to the server computer. The credential may be time limited and encrypted by the TTP using a symmetric algorithm and decrypted/verified by the server computer using the same. The TTP is used for a large aggregation of machines and contains all of the keys used for authentication by both users and machines.

[0005] A drawback with the TTP is that it requires significant infrastructure and is a separate entity that must be configured. Furthermore, the configuration information must be present on all of the server computers. Therefore, in order to implement a change, all of the machines (i.e. client and server computers), as well as the TTP, must be re-configured. Another drawback of the TTP is that it is a high value target because it contains all of the keys used for authentication.

[0006] Another type of authentication mechanism is pretty good privacy (PGP Ticket). A TTP (e.g., server administrator) issues a credential to a client computer that allows the server computer to authenticate the client computer. The credential issued to the client computer is time limited and digitally signed by the TTP using commonly known public key technology. The credential is interpreted/verified by the server computer.

[0007] A drawback with PGP Ticket is that the security and verifiability of the TTP's public key is weak. Furthermore, a change to the TTP's key requires revoking all of the old keys and updating all of the server computers and client computers with

new keys. Additionally, if the TTP's key is compromised, then all of the server computers that rely on that key are also compromised.

SUMMARY OF THE INVENTION

[0008] In accordance with the present invention there is provided a method for authenticating computers. The method comprises a first computer (e.g., a server computer) issuing a credential to a second computer (e.g., a client computer). When the second computer attempts to authenticate with the first computer, the second computer generates a first challenge and transmits the credential and the first challenge to the first computer. The first computer determines whether the credential is valid and computes a first response to the first challenge. Additionally, the first computer generates a second challenge. The first computer transmits the first response and the second challenge to the second computer. In order to authenticate the first computer, the second computer determines whether the first response is valid. The second computer also computes a second response to the second challenge. The second computer transmits the second response to the first computer which then determines whether the second response is valid in order to authenticate the second computer and establish a connection.

[0009] The credential may be encrypted before issuing it to the second computer and transmitting it from the second computer to the first computer. Similarly, the first challenge, the first response, the second challenge, and the second response may be encrypted before transmission. Each of the first challenges and responses, as well as the second challenges and responses, are decrypted upon receipt by the first or second computers.

[0010] The second challenge may be a random number such as a nonce generated by the first computer. The second computer computes a second response to the first computer challenge by performing a predetermined function on the random number. The first computer determines whether the second response is valid by performing the predetermined function on the random number and comparing the result to the second response. The predetermined function may be a hash function.

[0011] Similarly, the first challenge may be a random number such as a nonce. The first computer computes a first response to the first challenge by performing a predetermined function on the random number. The second computer determines whether the first response is valid by performing the predetermined function on the random number and comparing the result to the first response.

[0012] The credential may be issued with an expiration time. Once the expiration time has been reached, the credential is no longer valid.

[0013] In accordance with the present invention, there is provided a computer-readable medium containing a program with instructions to execute the authentication between a first computer and a second computer. The instructions issue a credential from the first computer to the second computer. The credential and a first challenge are transmitted from the second computer to the first computer when the second computer is to be authenticated. The first computer determines whether the credential is valid and computes a response to the first challenge. Furthermore, the first computer generates a second challenge that is transmitted to the second computer with the first response. The second computer determines whether the first response is valid and computes a second response to the second challenge. The second response is

transmitted to the first computer and the first computer determines whether the second response is valid in order to authenticate the second computer.

[0014] A system for authenticating computers has a first computer and a second computer in communication with each other. The first and second computers are configured to execute instructions which authenticate the second computer. Specifically, the first computer issues a credential to the second computer. In order to be authenticated, the second computer transmits the credential along with a first challenge to the first computer. The first computer is configured to determine whether the credential is valid and computes a first response to the first challenge. The first computer generates and transmits a second challenge and the first response to the second computer which verifies the first response and generates a second response. The second response is transmitted from the second computer to the first computer in order to authenticate the second computer.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] These, as well as other features of the present invention, will become more apparent upon reference to the drawings wherein:

[0016] FIG. 1 is a network diagram;

[0017] FIG. 2 is a flowchart illustrating how a reconnect credential is used by a server computer to authenticate a client computer;

[0018] FIG. 3 is a flowchart illustrating how a reconnect credential is issued by the server computer;

[0019] FIGS. 4 – 6 are flowcharts illustrating how a client computer uses the reconnect credential to authenticate with the server computer; and

[0020] FIG. 7 is a flowchart illustrating how a reconnect credential is reissued to a client computer.

DETAILED DESCRIPTION

[0021] Various aspects will now be described in connection with exemplary embodiments, including certain aspects described in terms of sequences of actions that can be performed by elements of a computer system. For example, it will be recognized that in each of the embodiments, the various actions can be performed by specialized circuits, circuitry (e.g., discrete and/or integrated logic gates interconnected to perform a specialized function), program instructions executed by one or more processors, or by any combination. Thus, the various aspects can be embodied in many different forms, and all such forms are contemplated to be within the scope of what is described. The instructions of a computer program as illustrated in FIGS. 2-7 for issuing and reconnecting with an authentication credential can be embodied in any computer-readable medium for use by or in connection with an instruction execution system, apparatus, or device, such as a computer based system, processor containing system, or other system that can fetch the instructions from a computer-readable medium, apparatus, or device and execute the instructions.

[0022] As used herein, a "computer-readable medium" can be any means that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device. The computer-readable medium can be, for example but is not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, device, or propagation medium. More specific examples (a non exhaustive list) of the

computer readable-medium can include the following: an electrical connection having one or more wires, a portable computer diskette, a random access memory (RAM), a read only memory (ROM), an erasable programmable read only memory (EPROM or Flash memory), an optical fiber, and a portable compact disc read only memory (CDROM).

[0023] Referring now to the drawings wherein the showings are for purposes of illustrating preferred embodiments of the present invention only, and not for purposes of limiting the same, Figure 1 illustrates client computers 12a, 12b, 12c connected to a network 18. Server computers 14a, 14b, and 14c are also connected to the network 18. In order for a client computer 12 to access a server computer 14, the client computer 12 transmits user identification and password information to the desired server 14. The server computer 14 uses this information to authenticate the client computer 12 and establish a connection.

[0024] Figure 2 is a flowchart showing how the reconnect credential is used is shown. In step 202, the server computer 14 issues a credential. Referring to Figure 3, one embodiment of a method of initially issuing the credential from the server computer 14 to the client computer 12 is shown. The client computer 12 already has an agreed upon initial session key (k1), while the server computer 14 has a long term server key (ks) and the initial session key (k1). In step 302, the client computer sends the initial timestamp (t1) encrypted using the initial session key (k1) to the server computer 14. In step 304, the server computer chooses security parameters and an expiration time (exp) for the credential. In the illustrated embodiment, the security parameters include a hash seed (s), and a maximum number of times (m) to run a hash function. In step 306, the server computer 14 calculates a time interval (t3) that is the

time interval between the clock of the server computer 14 and initial timestamp (t1). Information that uniquely identifies the session (sessioninfo) is generated in step 308 by the server computer 14 in order to distinguish the session from other sessions.

[0025] The server computer 14 generates a credential (cred) for the client computer 12 in step 310. As previously discussed, the credential proves the identity of the client computer 12. For the embodiment shown in Figure 3, the credential (cred) is generated using the long term server key (ks) and the seed (s), the maximum number of times (m) to run the hash function, the expiration time (exp), the time interval (t3), and username (user/domain) information to uniquely determine the client computer 12. It will be recognized by those of ordinary skill in the art that there may be other methods of generating the credential in order to prove the identity of the client computer 12. In step 312, the server computer 14 transmits the session information (sessioninfo), seed (s), maximum times (m) to run the hash function, the expiration time (exp), and the credential (cred) encrypted by the initial session key (k1) to the client computer 12 in order to issue the credential.

[0026] The credential received by the client computer 12 is used to re-establish a seamless connection with the server computer 14. In the reconnect procedure shown in Figure 2, the connection between the server computer 14 and the client computer 12 is terminated in step 204. The connection may be lost by unexpected session termination. In order to reconnect, the client computer 12 transmits the credential to the server computer 14 in step 206. In step 208, the process of authenticating the client computer 12 and server computer 14 with the credential issued from step 202 is performed. Once the authentication is complete, the connection is reestablished in step 210.

[0027] Referring to Figures 4A-4B, the method from step 208 for authenticating the client computer 12 and the server computer 14 with the credential issued in step 202 is shown. The client computer 12 has the credential (cred), seed (s), and maximum number of times (m) to perform the hash function. In step 402, the client computer 12 chooses the number of times (n) to run the hash function. Also, the client computer 12 chooses a client computer challenge such as a random number (client_nonce) for security purposes. The client computer 12 then signs the data using a hashed message authentication code (HMAC) and hashed seed (s) data in step 404. The data to be signed is the credential (cred), the client computer's random number (client_nonce), the number of times (n) to run the hash function, the timestamp (t2) and the seed (s) data hashed n and (n-1) times. The data signed by the HMAC is then transmitted from the client computer 12 to the server computer 14 in step 406.

[0028] In step 408, the server computer 14 checks the signatures and decrypts the credential (cred) using the long term server key (ks). If the decrypt fails, then the process exits in step 410. If the decrypt does not fail, then in step 412, the server computer 14 retrieves the seed (s), the maximum number of times (m) to perform the hash function, the expiration time of the credential (exp), and the time interval (t3).

[0029] The server computer 14 hashes the credential (cred) and determines if the credential (cred) is in a revocation list in step 414. If the hashed credential (cred) is in the revocation list, then the credential (cred) is not valid and the process exits in step 416. However, if the hashed credential (cred) is not in the revocation list, then the process proceeds to step 418 where the server computer 14 checks to see if the expiration time (exp) of the credential (cred) has been exceeded. If the time has been exceeded then the credential (cred) is not valid and the process exits in step 420.

Furthermore, in step 418, the server computer 14 determines whether the maximum number of times (m) to perform the hash function is greater than or equal to the number of times (n) to perform the hash function. If m is not greater than or equal to n then the process exits in step 420. Furthermore, the server computer 14 validates HMAC(s) and the user/domain in step 418. If either one of these values is invalid, then the process exits in step 420.

[0030] The server computer 14 performs two separate hashes. It hashes the seed (s) data n number of times ((s)H(n)) and n-1 times ((s)H(n-1)) in step 422. In step 424, the server computer 14 compares the values of (s)H(n) and (s)H(n-1) found in step 422 with the values decrypted in step 408. If the values do not match, then the process exits in step 426.

[0031] Next, in step 428, the server computer 14 decrypts and hashes the client_nonce to generate a server computer response. The server computer 14 then chooses a server computer challenge such as a random number (server_nonce) in step 430. In step 432, the server computer 14 adds the hashed credential (cred) to the revocation list so that it cannot be used again. In order to continue authentication of the credential, in step 434 the server computer 14 encrypts and transmits the server_nonce chosen in step 430, the hashed client_nonce from step 428 using as a key the seed data (s) hashed n-1 times.

[0032] Referring to Figure 5, the process of authenticating the client computer 12 is continued. Specifically, in step 502, the client computer 12 receives the message containing the server_nonce, the hashed client_nonce, encrypted using the seed data (s) hashed n-1 times as the key, transmitted by the server computer 14 in step 434. In step 504, the client computer 12 decrypts the message received in step 502. Next, in

step 506, the client computer 12 verifies the hashed client_nonce by comparing it with the known value. If the values do not match, then the process exits in step 508. However, if the values do match, then the client computer 12 can be assured that the server computer 14 is authentic because it successfully hashed the client_nonce. In step 510, the client computer 12 hashes the server_nonce received from step 502 to generate a client computer response. In order to verify to the server computer 14 that the client computer 12 has successfully received the message, the client computer 12 transmits the hashed server_nonce encrypted with the seed (s) hashed n-1 times to the server computer 12 in step 512.

[0033] Referring to Figure 6, the server computer 14 receives the message from step 512 and decrypts the message in step 604. The server computer 14 verifies the hashed server_nonce by comparing it to a known value in step 608. If the server_nonce does not match the value generated earlier by the server computer 14, then the process exits in step 609. However, if the server_nonce is verified, then both the server computer 14 and the client computer 12 are authenticated to each other and a connection can be established in step 610.

[0034] It is possible that the credential expires before being used by the client computer 12. As previously discussed, the credential includes an expiration time (exp) after which the credential cannot be used. If the credential is not used before the expiration time, the credential is invalid. Therefore, it is necessary to re-issue a valid credential to the client computer 12 before the end of the expiration time. Any reasonable rule can be used to determine when to reissue a credential prior to or after its expiration. As seen in Figure 2, the credential is about to expire in step 212.

Accordingly, in step 214, the server computer 14 re-issues the credential to the client computer 12.

[0035] Referring to Figure 7, one embodiment of a method for reissuing the credential is shown while the client computer 12 is still connected to the server computer 14. The client computer 12 has the credential (cred), seed (s) data, and initial session key (k1). The server computer 14 has the long term server key (ks), the initial session key (k1), and the unique name to identify the session (sessioninfo). In step 702, the client computer 12 transmits the credential (cred) and the new timestamp (t1) encrypted by the initial session key (k1) to the server computer 14. The server computer 14 receives and decrypts the message from the client computer 12 in step 704. Next, the server computer 14 determines if the credential (cred) has expired in step 706. If the time limit for the credential has expired, then the process exits in step 708. If the time limit has not expired, then the process proceeds to step 710 where both the client computer 12 and the server computer 14 compute a new session key (k1') that is equal to the hash of the credential (cred) and the seed (s). The server computer 14 creates a new credential (cred') in step 712 with a new seed (s'), new expiration time (exp') and new time interval (t3'). The server computer 14 places the old credential (cred) on the revocation list of invalid credentials in step 714. Once the old credential (cred) is on the revocation list, then the server computer 14 transmits the new credential (cred'), new seed (s'), new maximum times (m') to perform the hash function, the new expiration time (exp'), and the session information (sessioninfo) to the client computer 12 in step 716. The message from the server computer 14 to the client computer 12 is encrypted using the new session key (k1').

In step 718, the client computer 12 decrypts the message using the new session key (k1') to retrieve the new credential and values.

[0036] In order to provide added security, the long term server key (ks) can be changed. In order to change the long term server key (ks), the server computer 14 generates a cutoff date (cutoff_date) that is later than or equal to the expiration date (exp). Next, the server computer 14 generates a new long term server key (ks'). The cutoff date (cutoff_date), the original long term server key (ks), and the new long term server key (ks') are stored at the server computer 14. When attempting to decrypt a credential, the server computer 14 first uses the original long term server key (ks) and if that fails then uses the new long term server key (ks'). New credentials are issued using the new long term server key (ks'). When the cutoff date (cutoff_date) has passed, then the long term server key (ks) should be set to the new long term server key (ks') because all of the old credentials issued using the long term server key (ks) have expired. Also, once the cutoff date has passed, the revocation list of invalid credentials can be purged and the server computer 14 can issue a new long term server key using the method just described.

[0037] The reconnect authentication method described above provides security against common attacks. For example, because the client computer 12 and the server computer 14 transmit encrypted client_nonce and server_nonce which require the knowledge of the seed (s), the ability to stage a Man-in-the-Middle (MitM) attack where an attacker intercepts and possibly alters data traveling along the network is decreased. The method also provides security against reply attacks where an attacker bugs the network data and reuses it when needed because the messages between the client computer 12 and the server computer 14 include a timestamp which prevent

reuse at another time. Reflection attacks where an attacker re-transmits a message created by a user and obtains a right of access are thwarted by using chained nonces and having the user information (user/domain) in the credential. Furthermore, reflection attacks are reduced by having each message between the client computer 12 and server computer 14 be non-symmetric and using a unique long term server key (ks). The chained nonces also provide against interleaved attacks where an attacker disguises oneself under a different identity during communication. A chosen text attack where an attacker tampers with the message and attacks the cryptography algorithm itself is reduced by not using the long term server computer key (ks) to encrypt any data that is obtained from the client computer 12. Finally, forced delay attacks where an attacker intercepts the data in the protocol to use later are reduced by using timestamps and an expiration time for the credential, as well as using a revocation list.

[0038] It will be appreciated by those of ordinary skill in the art that the concepts and techniques described here can be embodied in various specific forms without departing from the essential characteristics thereof. For example, the authentication between the server computer 14 and the client computer 12 may be a single challenge whereby the client computer 12 does not need to retransmit any more information to the server computer 14 after sending the credential and challenge. The presently disclosed embodiments are considered in all respects to be illustrative and not restrictive. The scope of the invention is indicated by the appended claims, rather than the foregoing description, and all changes that come within the meaning and range of equivalents thereof are intended to be embraced.